listing subsets of omega in order

Johan G. F. Belinfante
2006 August 16

```
In[1]:= SetDirectory["l:"]; << goedel84.15b; << tools.m

        :Package Title: goedel84.15b       2006 August 15 at 7:20 p.m.

        It is now:  2006 Aug 16 at 4:34

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 August 15

        weightlimit = 40
```

summary

Any subset of the set **omega** can be listed in order. It follows from this that every subset of **omega** is either finite or countably infinite.

ordlist for subsets of omega

Lemma 1.

```
In[2]:= Map[implies[#, equal[x, range[ordlist[x]]]] &,
         SubstTest[and, subclass[u, v], subclass[v, u], {u → x, v → range[ordlist[x]]}]]

Out[2]= or[equal[x, range[ordlist[x]]], not[subclass[x, range[ordlist[x]]]]] == True

In[3]:= or[equal[x_, range[ordlist[x_]]], not[subclass[x_, range[ordlist[x_]]]]] := True
```

Lemma 2.

```
In[4]:= SubstTest[implies, and[subclass[y, omega], not[member[y, FINITE]]],
         equal[U[y], omega], y → range[ordlist[x]]]

Out[4]= or[equal[omega, U[range[ordlist[x]]]], member[intersection[OMEGA, x], FINITE],
          not[subclass[range[ordlist[x]], omega]]] == True

In[5]:= or[equal[omega, U[range[ordlist[x_]]]], member[intersection[OMEGA, x_], FINITE],
          not[subclass[range[ordlist[x_]], omega]]] := True
```

Lemma 3.

```
In[6]:= SubstTest[implies, equal[y, U[range[ordlist[x]]]],
         subclass[intersection[x, OMEGA, y], range[ordlist[x]]], y → omega]

Out[6]= or[not[equal[omega, U[range[ordlist[x]]]]],
          subclass[intersection[omega, x], range[ordlist[x]]]] == True

In[7]:= or[not[equal[omega, U[range[ordlist[x_]]]]],
          subclass[intersection[omega, x_], range[ordlist[x_]]]] := True
```

Theorem.

```
In[8]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
          implies[p1, p4], implies[and[p2, p4], p5], implies[and[p3, p5], p6],
          implies[p6, p7], implies[and[p1, p7], p8], implies[p8, p9],
          not[implies[and[p1, p2], p9]], {p1 → subclass[x, omega],
           p2 → not[member[x, FINITE]], p3 → subclass[range[ordlist[x]], omega],
           p4 → subclass[x, OMEGA], p5 → not[member[intersection[OMEGA, x], FINITE]],
           p6 → equal[U[range[ordlist[x]]], omega],
           p7 → subclass[intersection[omega, x], range[ordlist[x]]],
           p8 -> subclass[x, range[ordlist[x]]], p9 → equal[x, range[ordlist[x]]]}]]

Out[8]= or[equal[x, range[ordlist[x]]], member[x, FINITE], not[subclass[x, omega]]] == True

In[9]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary.

```
In[10]:= Map[not, SubstTest[and, implies[p1, p3], implies[p2, p4], implies[and[p3, p4], p5],
           not[implies[p1, p5]], {p1 → subclass[x, omega], p2 → member[x, FINITE],
            p3 → subclass[x, OMEGA], p4 → equal[range[ordlist[x]], intersection[x, OMEGA]],
            p5 → equal[x, range[ordlist[x]]]}]]

Out[10]= or[equal[x, range[ordlist[x]]], not[subclass[x, omega]]] == True

In[11]:= or[equal[x_, range[ordlist[x_]]], not[subclass[x_, omega]]] := True
```

cardinality of subsets of omega

Theorem. Every set of natural numbers is either finite or countably infinite.

```
In[12]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p2, or[p3, p4]], not[implies[p1, or[p3, p4]]],
           {p1 → subclass[x, omega], p2 -> equal[x, range[ordlist[x]]],
            p3 → member[x, FINITE], p4 → equal[omega, card[x]]}]]

Out[12]= or[equal[omega, card[x]], member[x, FINITE], not[subclass[x, omega]]] == True

In[13]:= or[equal[omega, card[x_]], member[x_, FINITE], not[subclass[x_, omega]]] := True
```

Corollary. (Variable-free reformulation.)

```
In[14]:= Map[equal[V, #] &,
           SubstTest[class, x, implies[member[x, u], or[member[x, v], member[x, w]]],
            {u → P[omega], v → FINITE, w -> image[Q, set[omega]]}]] // Reverse

Out[14]= subclass[P[omega], image[Q, succ[omega]]] == True

In[15]:= subclass[P[omega], image[Q, succ[omega]]] := True
```

Lemma.

```
In[16]:= ImageComp[CARD, Q, succ[omega]] // Reverse

Out[16]= image[CARD, image[Q, succ[omega]]] == succ[omega]

In[17]:= image[CARD, image[Q, succ[omega]]] := succ[omega]
```

Lemma.

```
In[22]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u -> P[omega], v -> image[Q, succ[omega]], w → Q}]

Out[22]= subclass[image[Q, P[omega]], image[Q, succ[omega]]] == True

In[23]:= % /. Equal → SetDelayed
```

Theorem.

```
In[24]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> image[Q, P[omega]], v → image[Q, succ[omega]]}]

Out[24]= True == equal[image[Q, P[omega]], image[Q, succ[omega]]]

In[25]:= image[Q, P[omega]] := image[Q, succ[omega]]
```

Theorem.

```
In[26]:= ImageComp[CARD, Q, P[omega]]

Out[26]= image[CARD, P[omega]] == succ[omega]

In[27]:= image[CARD, P[omega]] := succ[omega]
```

U rules for even and odd

The set of even numbers is unbounded.

```
In[28]:= SubstTest[U, image[DIV, set[x]], x → succ[set[0]]]

Out[28]= U[even] == omega

In[29]:= U[even] := omega
```

So is the set of odd numbers.

```
In[30]:= Map[equal[#, omega] &, SubstTest[U, image[SUCC, x], x → even]]

Out[30]= equal[omega, U[odd]] == True

In[31]:= U[odd] := omega
```

even and odd are countably infinite

Corollary.

```
In[32]:= Map[not, SubstTest[member, x, intersection[y, z],
            {y -> image[inverse[BIGCUP], set[omega]], z -> P[omega]}]] /. x → even

Out[32]= member[even, FINITE] == False

In[33]:= member[even, FINITE] := False

In[34]:= Map[not, SubstTest[member, x, intersection[y, z],
            {y -> image[inverse[BIGCUP], set[omega]], z -> P[omega]}]] /. x → odd

Out[34]= member[odd, FINITE] == False

In[35]:= member[odd, FINITE] := False

In[36]:= SubstTest[implies, and[subclass[x, omega], not[member[x, FINITE]]],
          equal[omega, card[x]], x → even]

Out[36]= equal[omega, card[even]] == True

In[37]:= card[even] := omega

In[38]:= SubstTest[implies, and[subclass[x, omega], not[member[x, FINITE]]],
          equal[omega, card[x]], x → odd]

Out[38]= equal[omega, card[odd]] == True

In[39]:= card[odd] := omega
```

A rules for even and odd

This is automatic, so no new rewrite rule is needed.

```
In[40]:= A[even]

Out[40]= 0
```

The least odd number is **1**.

```
In[41]:= SubstTest[implies, member[x, y], subclass[A[y], x], {x → set[0], y → odd}]

Out[41]= subclass[A[odd], set[0]] == True

In[42]:= % /. Equal → SetDelayed

In[43]:= SubstTest[implies, subclass[u, v],
          subclass[A[v], A[u]], {u → odd, v → dif[omega, set[0]]}]

Out[43]= member[0, A[odd]] == True

In[44]:= % /. Equal → SetDelayed

In[45]:= equal[A[odd], set[0]]

Out[45]= True

In[46]:= A[odd] := set[0]
```

---

other card rules

Some other cardinality rewrite rules.

```
In[47]:= SubstTest[implies, FUNCTION[x], equal[card[x], card[domain[x]]], x → FACTORIAL]

Out[47]= equal[omega, card[FACTORIAL]] == True

In[48]:= card[FACTORIAL] := omega

In[49]:= SubstTest[implies, FUNCTION[x], equal[card[x], card[domain[x]]], x → PRIMESEQ]

Out[49]= equal[omega, card[PRIMESEQ]] == True

In[50]:= card[PRIMESEQ] := omega
```